

WHAT IS CLAIMED IS:

1 1. A method of managing messages, comprising:
2 storing messages in a plurality of queues;
3 providing a macro queue associated with the plurality of queues;
4 calling an application programming interface to initiate a request to the macro queue
5 to obtain a message stored in one of the plurality of queues without identifying a particular
6 queue; and
7 selecting a queue from among the plurality of queues and selecting a message from
8 the selected queue.

1 2. The method of claim 1, further comprising assigning a priority value to each
2 of the plurality of queues.

1 3. The method of claim 2 wherein the macro queue selects a message from a
2 queue having the highest priority value.

1 4. The method of claim 1 wherein the macro queue selects a message that has
2 been stored in the plurality of queues for the longest time.

1 5. The method of claim 1, further comprising providing a remote queue proxy
2 for establishing a communication link between a remote application programming interface
3 and the macro queue.

1 6. The method of claim 1 wherein the plurality of queues and the macro queue
2 are software objects that are implemented using object oriented programming principles.

1 7. The method of claim 6, further comprising calling a software function of the
2 macro queue object to associate a queue object with the macro queue object, wherein the
3 software function returns a queue instance pointer pointing to the location of the queue object
4 and a priority value representing the priority of the queue.

1 8. The method of claim 6, further comprising calling a software function of the
2 macro queue object to remove the association between the macro queue and a queue.

1 9. A method of managing messages, comprising:
2 providing an application programming interface (API) to allow a producer module to
3 send a message to a macro queue that manages a plurality of queues, the API sending the
4 message to the macro queue without identifying one of the plurality of queues.

1 10. The method of claim 9 wherein the macro queue selects the first queue that is
2 available among the plurality of queues and sends the message to the selected queue.

1 11. The method of claim 9 wherein the macro queue duplicates the message and
2 sends the message to all of the plurality of queues.

1 12. The method of claim 9 wherein the macro queue selects a queue from among
2 the plurality of queues that has the fastest response time based on previous response time
3 records and sends the message to the selected queue.

1 13. The method of claim 9 wherein the macro queue selects a queue by cycling
2 through each of the plurality of queues in a round robin fashion, and sends the message to the
3 selected queue.

1 14. The method of claim 9 wherein the macro queue and the plurality of queues
2 are implemented as software objects according to objected oriented programming principles.

1 15. A method comprising:
2 keeping a list of queue pointers, each pointer pointing to one of a plurality of queues;
3 receiving a request for adding a queue element; and
4 servicing the request by selecting one or more queue pointers from the list based on a
5 predetermined criterion and adding the queue element to the one or more queues that the
6 selected one or more queue pointers are pointing to.

1 16. The method of claim 15 wherein the predetermined criterion is to select a
2 queue pointer pointing to a queue that has the shortest response time.

1 17. The method of claim 15 wherein the predetermined criterion is to select all of
2 the queue pointers.

1 18. The method of claim 15 wherein the predetermined criterion is to select a
2 queue pointer from the list in a round robin fashion by cycling through each of the queue
3 pointers in the list.

1 19. A method comprising:
2 keeping a list of queue pointers, each pointer pointing to one of a plurality of queues;
3 receiving a request for retrieving a queue element; and
4 servicing the request by selecting one or more queue pointers from the list based on a
5 predetermined criterion and retrieving a queue element from the one or more queues that the
6 selected one or more queue pointers are pointing to.

1 20. The method of claim 19 wherein the predetermined criterion is to select a
2 queue pointer pointing to a queue that is the first one to be available.

1 21. The method of claim 19 wherein each of the queues has a priority value, and
2 the predetermined criterion is to select a queue pointers pointing to a queue having the
3 highest priority value.

1 22. A method for messages communication in a distributed system, comprising:
2 providing an application programming interface on each computer of a group of
3 computers in the distributed system;
4 providing a remote queue proxy on each of the computers of the group;
5 initiating a request through an application programming interface on a first computer
6 of the group; and
7 passing said request to a second of the computers of the group by passing said request
8 through the remote queue proxy on the first computer and the remote member queue proxy
9 on said second computer.

1 23. The method of claim 22 wherein providing the application programming interface
2 includes providing software objects implementing said interface that are implemented using
3 object oriented programming principles.

1 24. The method of claim 22 wherein providing the remote queue proxy includes
2 providing a software object implementing said proxy.

1 25. A method for passing messages between processes in a distributed system
2 comprising:
3 providing an application programming interface to processes hosted on computers of
4 the distributed system;
5 passing a first message from a first process to a second process hosted on one
6 computer of the distributed system, including passing said message through a shared memory
7 accessible to both the first process and the second process; and
8 passing a second message from the first process to a third process hosted on a second
9 computer of the distributed system, including passing said message over a communication
10 channel coupling the first and the second computers.

1 26. The method of claim 22 wherein the first process uses the same application
2 programming interface to pass the first message and the second message.

1 27. The method of claim 22 wherein the first process is unaware of whether the first
2 message and the second message are passing to a process hosted on the first computer or the
3 second computer.

1 28. The method of claim 22 wherein providing the application programming interface
2 includes providing a queuing interface for passing messages between computers.

1 29. The method of claim 22 further comprising:
2 providing a macro queue associated with the plurality of queues; and
3 wherein passing the first message from the first process to the second process
4 includes calling the application programming interface to initiate a request to the macro
5 queue to obtain a message stored in one of the plurality of queues without identifying a
6 particular queue and selecting a queue from among the plurality of queues and selecting a
7 message from the selected queue.

1 30. The method of claim 22 further comprising:
2 providing a remote queue proxy for establishing the communication channel between
3 the first and the second computers.

1 31. A method for message passing in a distributed system comprising:
2 providing a queue manager on each of a group of computers in the distributed system;
3 providing an application programming interface to processes on each of the
4 computers of the group, including providing an interface to accept and to provide messages
5 for passing between processes hosted on the computers;
6 collecting operational statistics at each of the queue managers related to passing of
7 messages between processes using the application programming interface; and
8 optimizing passing of the messages according to the collected statistics.

1 32. A method for fault-tolerant operation of a system comprising:
2 providing redundant processes for processing messages;
3 providing a separate replicated message queue for each of the redundant processes;
4 accepting a message for processing by each of the redundant processes;
5 enqueueing the message into each of the replicated message queues such that the order
6 of message dequeuing from said queues by the redundant processes is synchronized.

1 33. The method of claim 22 wherein enqueueing the message into each of the message
2 queues includes performing a logically atomic enqueueing operation on all the queues.

1 34. The method of claim 22 wherein providing each of said replicated queues
2 includes providing a replicated macro queue associated with a plurality of replicated member
3 queues of said macro queue.

1 35. A method of managing messages, comprising:
2 providing an application programming interface (API) to allow a producer
3 module to send a message to a macro queue that manages a plurality of member
4 queues, the API sending the message to the macro queue without identifying one of
5 the plurality of member queues; and

6 using the same API to allow the producer module to send a message to an
7 individual queue.

8
1 36. The method of claim 35 wherein the macro queue selects one or more of the
2 member queues according to a predefined criteria.

3
1 37. The method of claim 36 wherein the macro queue, the member queues, and
2 the individual queue are implemented as software objects according to object oriented
3 programming principles.

1 38. A method of managing messages, comprising:
2 providing an application programming interface (API) to allow a consumer
3 module to retrieve a message from a macro queue that manages a plurality of member
4 queues, the API retrieving the message from the macro queue without identifying one
5 of the plurality of member queues; and
6 using the same API to allow the consumer module to retrieve a message from
7 an individual queue.

8
1 39. The method of claim 38 wherein the macro queue selects one of the member
2 queues according to a predefined criteria and selects a message from the selected
3 member queue.

4
1 40. The method of claim 39 wherein the macro queue, the member queues, and
2 the individual queue are implemented as software objects according to object oriented
3 programming principles.
4
5